云存储中基于 MHT 的动态数据完整性验证与恢复方案 *

李敬伟 [1]，朱命冬 [1, 2]

(1. 河南工学院 计算机科学与技术系，河南 新乡 453002; 2. 东北大学 计算机科学与工程学院，沈阳 110006)

摘 要：针对云服务器上存储数据完整性验证过程中的高通信开销和动态数据验证问题，提出一种基于 Merkle 哈希树(MHT)的动态数据完整性验证与恢复方案。首先，基于 MHT 构建了一种新型分层认证数据结构，将数据块的每个副本块组织成副本子树，以此大幅降低多副本更新验证的通信开销。然后，在数据验证中，融入了对服务器安全索引信息的认证，以此避免服务器攻击。最后，当发现数据损坏时，通过二分查找和 Shamir 秘密共享机制来恢复数据。实验结果表明，该方案在验证过程中能有效降低计算和通讯开销，并能够很好地支持数据的动态操作。

关键词：云存储；数据完整性验证；Merkle 哈希树；动态数据；数据恢复

中图分类号：TP309.2　　doi: 10.3969/j.issn.1001-3695.2018.01.0097

Dynamic data integrity verification and recovery scheme in cloud storage based on MHT

Li Jingwei[1], Zhu Mingdong[1, 2]

(1. *Dept. of Computer Science Henan Institute of Technology*, *Xinxiang Henan* 453002, *China*; 2. *College of Computer Science & Engineering*, *Northeastern University*, *Shenyang* 110006, *China*)

Abstract: For the issues that the high communication overhead and dynamic data verification in the storage data integrity verification process on the cloud server, this paper proposed a dynamic data integrity verification and recovery scheme based on Merkle hash tree (MHT) . Firstly, it constructed a new layered authentication data structure based on MHT, and organized each copy block of the data block into a copy sub-tree, thereby greatly reducing the communication overhead of multi-copy update verification. Then, it certified the server security index information authentication into the data validation, in order to avoid server attacks. Finally, it recovered the data by dichotomous discovery and Shamir's secret sharing mechanism when the data is corrupted. The experimental results show that this scheme can effectively reduce the computational and communication overhead and can support the dynamic operation of data in the verification process.

Key words: cloud storage; data integrity verification; Merkle hash tree; dynamic data; data recovery

0 引言

云计算是新一代的分布式计算平台，对于大数据的存储和处理提供了很大便利[1]。云存储的安全性是云计算主要关心的问题之一，安全性的三个主要方面是机密性、完整性和可用性[2]。存储在云存储服务器(CSP)上的大数据集本质上是动态的，数据更新频繁。因此，使一些公共审计机制等云安全机制支持动态数据具有重要意义[3]。

公共审计机制是用来对外部数据进行完整性验证。凭借可证明数据持有(PDP)和可检索证明(POR)[4]，数据拥有者或第三方审计人员(TPA)可以验证其数据的完整性，而无需检索其数据[5]。现有的公共审计机制已经可以支持对动态更新数据进行验证，例如文献[6]使用了一种基于层级的认证跳表技术(RASL)作为认证数据结构，提出了一种支持动态数据更新验证的 PDP 方案。但是，可变大小的文件块在其框架中不受支持。文献[7]提出了一种基于 BLS 签名和 Merkle 哈希树认证数据结构的方案，可以支持公共审计和数据动态，但其不能抵御服务器伪造认证攻击。文献[8]同样通过基于 Merkle 哈希树认证数据结构中的辅助认证信息(AAI)来支持这种动态审计。

但是，这些方法仍然存在一些问题。首先，现有的研究缺乏对维护多个副本的动态数据集进行有效公共审计。存储多个副本是远程云存储中提高可靠性和可用性的常用策略[9]。对于高度动态的数据，每次更新都会导致更新每个副本。考虑到当前审计方案中的更新验证具有 $O(log\ n)$ 通信复杂度，逐个验证这些副本在通信方面将是非常昂贵的。其次，目前的动态公共审计方案由于缺乏块索引认证，容易受到恶意服务器的攻击。

为此，本文提出了一个基于 Merkle 哈希树(MHT)的多副本动态公共审计方案，通过一个新设计的认证数据结构解决上述问题。本文的主要研究贡献如下：

a）为了解决多副本云存储种数据更新的验证效率问题，提

收稿日期：2018-01-15；**修回日期：**2018-04-02　**基金项目：**国家自然科学基金资助项目（61762086）；河南省高等学校重点科研项目（16B520009）；河南省高等学校重点科研项目（18B520011）

作者简介：李敬伟（1982-），男，河南驻马店人，讲师，硕士，主要研究方向为大数据、云计算等（lijingweixx@126.com）；朱命冬（1985-），男，河南新乡人，讲师，博士，主要研究方向为大数据、计算机应用等.

出了一种基于 Merkle 哈希树的多副本公共审计方案，其中每个数据块的所有副本块被组织成相同的副本子树，以此大幅降低更新验证的通信开销。同时，在基于 Merkle 哈希树在动态数据公开审计中，融入了对服务器安全索引信息的认证，以此避免服务器攻击。

b）设计了一种能够一次性验证所有副本的验证机制，且在服务器端也可以减少额外的存储开销。

c）当完整性验证失败时，融入了损坏数据块定位和恢复过程，即通过二分查找来定位损坏数据块，通过 Shamir 秘密共享机制来恢复数据。

1　问题描述与分析

1.1　多副本验证

存储多个副本是云存储服务商的默认设置，将副本存储在不同的服务器，以便用来将用户数据从服务故障中恢复。用户验证多个副本完整性的直接方法是将它们作为单独的文件存储并逐一验证。目前，用于支持副本数据动态修改的最常用技术是构建一种支持更新的认证数据结构(ADS)。对于 ADS 的存储复杂性为 $\log(n)$，n 为块的总数，当文件很大时其非常大。更重要的是通信复杂性，当每个数据块发生更新时将需要更新每个副本中相应的块[10]，这将造成高通信成本。本文试图用一个新的 ADS 解决这个问题，将每个块的所有副本链接在一起。

文献[11]中提出了一个多副本验证方案，命名为 MR-PDP，将每个块和所有副本块只关联一个同态线性验证器(HLA)，效率较高。虽然这种方法可以带来很多好处，如服务器端存储成本较低，客户端预处理时间较短，但用第三方审计人员验证时，存在安全性问题。验证过程需要私有的随机数 $r_{i,j}$，如果被泄露，另一方将知道如何计算基于任何副本的原始消息，以及如何基于原始文件块来计算任意副本。更糟的是，如果 $r_{i,j}$ 被云服务器所知，则云服务器将能够伪造任何副本块的完整性证明。

总而言之，从本文的考虑，多副本验证方案应同时包括以下内容：

a）能够支持公开审计和动态数据更新。使第三方审计人员能够在不需要任何秘密信息的情况下为客户进行常规验证，并允许客户验证数据更新。

b）能够支持全面验证。可以一次对所有的副本进行有效的验证，如果任何副本失败，服务器会及时通知。

c）能够支持单副本验证。可以对某些特定块的任意副本进行验证，因为验证者可能只想知道对于不重要的数据是否至少有一个副本是完整的。

1.2　安全的动态公共审计

图 1 显示了客户/数据拥有者(DO)、云服务提供商(CSP)和第三方审计者(TPA)在公开审计中的关系。其中客户授权 TPA 审核存储在 CSP 上的数据，三方之间并不完全信任。

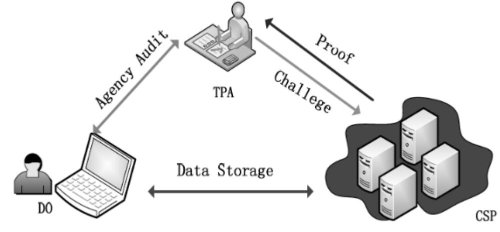


图 1 云数据公共审计中参与方的关系

MHT 或 RASL 等认证数据结构可以支持第三方审计者验证数据块的内容和更新[12]。对块索引进行验证可以避免恶意服务器根据另一个完整的块和其 AAI 来伪造证明。其次，对于动态数据的审计，在验证器计算中需要块的本身哈希值，而不是包含块索引的任何哈希值，例如 $H(i)$ 或 $H(v\|i)$，否则插入/删除操作将导致所有下层块改变。因此，每个验证者都将具有一个哈希值 $H(m_i)$。作为客户验证 $H(m_i)$ 正确性的唯一方法是通过 ADS，服务器可以用另一个哈希和 AAI 来欺骗客户端。换句话说，服务器可以采取任何其他的完整块来代替应该被验证的块，这就影响了完整性验证的结果。

文献[13]提出的 RASL 方法可以为索引提供认证，这对上述攻击具有抵抗性。其中认证者 T 被计算为 $T=g^m$，g 是生成者，m 是要审核的消息。但是 RASL 不能直接应用到支持动态数据的公共审计方案中。如前所述，消息块 $H(m_i)$ 的哈希值将被用在验证者中以支持动态数据。因此，客户需要通过云服务器的计算来验证。为了实现索引信息的可验证性，叶节点不再存储文件块的哈希值，而是以 $f(v)=H(l(v),r(v),x(v),f(rgt(v)))$ 的形式连接多个哈希值。因此，服务器需要发回 $f(v)$ 和 $H(m)$ 给客户端进行认证。

RASL 中存在一个缺陷，即当一条验证路径上具有多个叶节点。在这种情况下，如果需要验证一个数据库，则服务器不仅需要返回该数据块上的所有三个值，而且还需要计算和传递所有相同路径上数据库的验证值。为此，本文选择 Merkle 哈希树来构建 ADS。

2　双线性配对和 Merkle 哈希树

2.1　符号定义

文中所涉及的一些符号及其含义描述如表 1 所示。

表 1　文中所涉及的符号及含义

符号	含义
F	客户端上传的原始数据文件，存储在 CSP 中。
m_i	F 的第 i 个文件块，总共有 n 个块。
$\tilde{F}_j$	文件 F 的第 j 个副本。
$b_{i,j}$	副本 $\tilde{F}_j$ 的第 i 个块。
$r_{i,j}$	用于生成副本块 $b_{i,j}$ 的填充消息与原始文件块 m_i 的关系。
T	基于 m_i 开发的 $MR\text{-}MHT$。
T_i	基于 m_i 的 T 的副本-子树。
$H(m)$	消息 m 的哈希值。
$h(v)$	存储在节点 v 中来自于从 $MR\text{-}MHT$ T 的值。
$l(v)$	节点 v 的层级。
$r(v)$	可以从 v 到达叶子层中的最大节点数量。
$\sigma_{i,j}$	$b_{i,j}$ 的同态标签。
s	每块的段数。

Γ_i	用于 RST T_i 中的所有中间节点的元组 $\{h,l,q,t\}$。
sig_{AUTH}	用于授权 TPA 的签名。
Ω_i	用作 m_i 的辅助认证信息的元组。
R	存储在 T 中根节点上的哈希值。
(f_k,η_k,q_k,d_k)	Ω_i 中的第 k 个元组，其中 f_k 是哈希值，l_k 是节点的层级，q_k 是层级值，d_k 指示该节点是左子节点还是右子节点。
$(\lambda_k,\eta_k,\xi_k,\zeta_k)$	用于验证的变量元组。对于成功的验证，在用 Ω_i 进行迭代计算之后，λ 将成为总文件块的数量，η 将成为根值 R，ξ 将成为块索引，ζ 将成为反向块索引，即从右边的块数。

2.2 双线性配对

双线性配对在用于建立和验证同态验证器的公共审计方案中是必不可少的[14]。假设一个群 G 是一个间隙 Diffie-Hellman(GDH)群，它的质数为 p 。双线性映射是一个构造 $e:G\times G\rightarrow G_T$ 的映射，其中 G_T 是具有素数的乘法循环群。一个有用的双线性映射 e 应该具有以下属性：

a）双线性，$\forall m,n\in G\Rightarrow e(m^a,n^b)\Rightarrow e(m,n)^{ab}$；

b）非退化性，$\forall m\in G,m\neq 0\Rightarrow e(m,n)\neq 1$；

c）可计算性，e 应该是有效可计算的；

d）安全性，计算 G_T 中的离散对数问题是困难的。

2.3 Merkle 哈希树

Merkle 哈希树类似于二叉树，在叶子节点存储数据信息，非叶子节点的值是通过对其叶子节点值进行哈希运算得到，从底层向上层逐步运算，得到根节点值，用来描述数据信息的完整性[15]。每个节点 N 将具有最多两个子节点。MHT T 上的一个节点 N 中的信息被构造如下：对于包含文件块 m_i 的叶节点，节点值计算为 $h_i=H(m_i)$；N_1 和 N_2 的父节点被构造为 $N_p=\{H(h_1\|h_2)\}$。叶子节点 m_i 的辅助认证信息 Ω_i 是其上级节点选择的一组哈希值，这样就可以通过 (m_i,Ω_i) 计算根值 R 。MHT 认证数据的结构如图 2 所示。

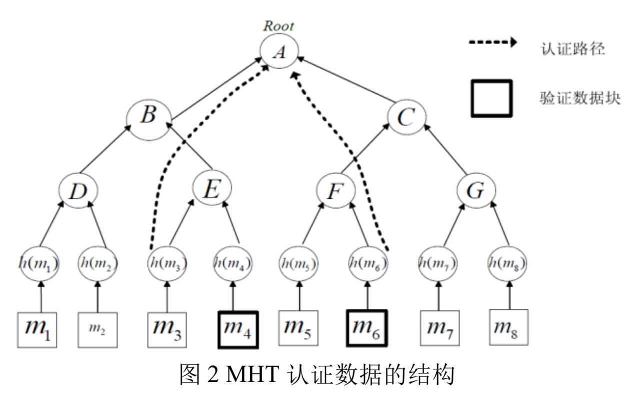


图 2 MHT 认证数据的结构

3 提出的多副本动态数据验证方案

3.1 多副本 Merkle 哈希树结构

多副本 Merkle 哈希树是一种新颖的认证数据结构，专为数据更新以及对块索引进行验证而设计。基于具有 3 个块和 1 个副本文件构建的一个多副本 MHT 的示例如图 3 所示。

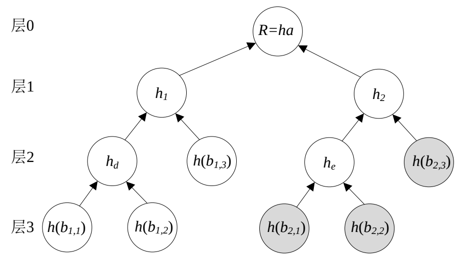


图 3 一个多副本 MHT 的例子

多副本 MHT 与传统 MHT 的差异如下：

a）存储在叶节点中的值是所存储的副本块的哈希值。在多副本 MHT 中，叶节点代表副本块 $b_{i,j}$，即第 i 个文件块的第 j 个副本。

b）存储在非叶节点 v 中的值是由其子节点的哈希值和另外两个索引 $l(v)$ 和 $r(v)$ 计算得来的。$l(v)$ 是节点 v 的层级，$r(v)$ 是 v 可以到达的叶子节点的最大数目。这些层级是从上到下的顺序定义的，根节点 R 的层级定义为 0，其子节点的层级定义为 1。叶子节点中存储的值为 $H(1\|l(b_{i,j})\|H(b_{i,j}))$；在每个非叶节点中的值为 $H(r\|l\|h_{left}\|h_{right})$，其中 h_{left} 和 h_{right} 分别表示存储在其左子节点和右子节点中的值。在图 3 中，根据本文的定义，$l(b_{i,j})$(对于所有叶节点)是 3，$r(b_{i,j})$=1。例如，h_d 的计算公式如下：

$$h_d=H(r(d)\|l(d)\|h(b_{1,1})\|h(b_{1,2}))$$
$$=H(2\|3\|h(b_{1,1})\|h(b_{1,2}))$$

且 $h(b_{1,2})=H(1\|4\|H(b_{1,2}))$，$h_b=H(6\|1\|h_1\|h_2)$ 等。

c）其中的 AAI $V_{i,j}$ 与 MHT 不同。每个节点上包含了一组信息 $\{h,l,q,d\}$。h 为存储在该节点上的哈希值，l 为该节点的层级，q 为从该节点可到达叶节点的最大数量，d 用来指示该节点到根节点 R 是向右(0)或向左(1)。例如，在图 3 中，对于副本块 $b_{2,1}$，$\Omega_{2,1}$ 被定义为 $\left\{\begin{matrix}(h(b_{2,1}),4,1,0),(h(b_{2,2}),4,1,0),\\(h(b_{2,3}),3,1,0),(h_2,1,6,0)\end{matrix}\right\}$，它的验证路径为 $\{h(b_{2,1}),h(e),h(2),R\}$。

d）一个文件块的所有副本被组织成一个相同结构的副本子树(RST)。子树中的叶节点数量为总副本数量 c 。副本块是独立

处理的，每个副本块都有自己的验证器。每个 RST 的根，表示为 h_i，将在多副本验证和更新验证中发挥重要作用。本文使用 Ω_i 来表示 h_i 的 AAI。另外，本文将 Γ_i 定义为每个 RST 中所有中间节点的元组 $\{h,l,q,t\}$ 的集合，其中 t 是节点的序号。由于副本数量较少(小于 10)，为了简化描述，本文假设 T_i 结构存储在客户端和 TPA 端，这适用于每个 RST，并且只占用可忽略的存储量。在这种情况下，客户端可以计算出 Γ_i，因此可以根据 $h(b_{i,j})$ 和 $l(b_{i,j})$ 来计算 Γ_i，而不需要从服务器请求 Γ_i。

3.2 初始化设置过程

用户和云服务器将首先建立通用参数，包括双线性映射 $e:G\times G\rightarrow G_T$ 和一个密码哈希函数 H [16]。

产生密钥操作 $KeyGen(1^k)$：客户端生成一个秘密值 $\alpha\in Z_p$ 和一个 G 的生成器 g，然后计算 $v=g^a$，其中 v、g 是公钥，α 是密钥。另一个签名密钥对 $\{spk,ssk\}$ 是根据指定的签名方案来选择的，其签名算法被表示为 $Sig()$。该算法输出 $\{ssk,\alpha\}$ 作为密钥 sk，$\{spk,v,g\}$ 作为公钥 pk。

文件预处理操作 $FilePreProc(F,sk,c)$，其分为以下三个步骤：

a）为了将数据集存储在云服务器上，客户端将首先根据原始文件制作 c 个副本。为了能够验证这些副本，它们应该是彼此不同的，否则服务器可能会用正确的证据来回应挑战，从而欺骗客户，但实际上只存储一个副本。原始文件为 $F=\{m_1,m_2,\ldots,m_n\}$，本文将其第 j 个副本文件表示为 $\tilde{F}_j=\{b_{1,j},b_{2,j},\ldots,b_{n,j}\}$，$j\in[1,c]$。副本中的块 $b_{i,j}$ 从 m_i 转换而来，并且转换是可逆的，即客户可以通过任何副本 $\tilde{F}_j$ 来恢复原始文件 F。例如，可以选择 n 个伪随机函数 ψ 来计算随机值 $r_{i,j}=\psi(j\|i)$，然后输出 $b_{i,j}=m_i+r_{i,j}$。

b）客户端基于 $b_{i,j}$ 构造一个多副本 MHT，计算根值 R，并用 ssk 计算其签名 sig。

c）对于每个副本块 $b_{i,j}$，客户端将计算一个认证者 $\sigma_{i,j}=\left(H(b_{i,j})\mu^{b_{i,j}}\right)^{\alpha}$。

最后，初始化过程输出 $\{b_{i,j},\sigma_{i,j},sig\}$，并上传到云服务器。

3.3 数据更新和验证过程

数据动态更新类型有全块插入(I)，删除(D)和修改(M)。在多副本场景中，当需要更新块 m_i 时，也需要以相同的方式更新其所有对应的副本块 $b_{i,j}$ 以保持一致性。同时客户端将计算新的副本块 $b'_{i,j}$，然后将它们与更新类型(I、D 或 M)一起发送到服务器。

数据更新操作 $PerformUpdate(UpdateReq)$：

服务器将 $UpdateReq$ 解析为 $\{Type,i,\{b'_{i,j}\}\}$。并根据更新请求对文件块、索引和 ADS 进行更新。请注意，Ω_i 中非叶节点中的值在更新过程后保持不变。

对于插入和删除，相邻 RST 中的所有叶节点的层级也被改变，其中插入操作时+1，删除操作时-1。

例如在图 4 中，随着 $\{b'_{2,j}\}$ 的插入，$\{b_{2,j}\}$ 的层级增加了 1，这将导致所有 $\{h(b_{2,j})\}$ 的改变；如果再将插入的块 $\{b_{2,j}\}$ 删除，旧的 $\{b'_{2,j}\}$ 的层级减少了 1。

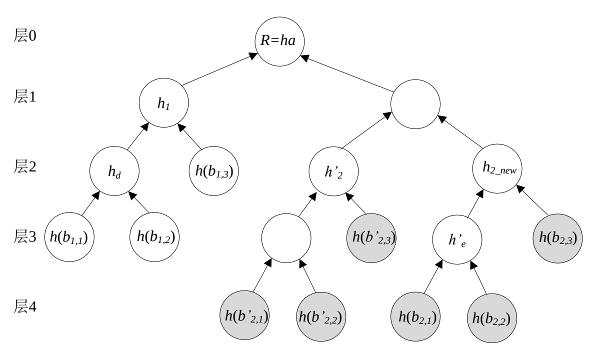


图 4 在第 2 个副本前插入块。

对于插入和修改操作，服务器计算 $P_{update}=\{\{h(b_{i,j})\},\Omega_i,R',sig\}$ 并将其返回给客户端。对于删除，服务器将需要另外发送 $h'(b_{i,j})$。

验证更新操作 $VerifyUpdate(pk,P_{update})$：

为了验证这个更新，客户端首先需要解析 P_{update}。令 Ω_i 中的 π 元组为 (f_k,l_k,q_k,d_k)，对于每个节点 N_k 按降序排列，即 $l_1=l(h_i),\ldots,l_{\pi-1}=2,l_\pi=1$。与先前定义有一点不同，$q_k$ 为可以从 N_k 到达的 RST 根的最大数目，而不是叶节点。由于 RST 的结构对于客户是已知的，将能够分别计算 h_i 和 h'_i，并根据 $h(b_{i,j})$(从服务器获得)和 $h'(b_{i,j})$ 计算 T_i 的新根和旧根。

a）对于在 Ω_i 中包含节点 N_k 的验证路径上的节点，客户端将首先迭代计算元组 $(\lambda_k,\eta_k,\xi_k,\zeta_k)$，其中 $k=1,\ldots,\pi$：如果 $d_k=1:\lambda_k=q_{k-1}+\lambda_{k-1}$、$\eta_k=H(\lambda_k\|l_k\|f_k\|\eta_{k-1})$、$\xi_k=\xi_{k-1}+q_k$ 和 $\zeta_k=\xi_{k-1}$。或者：如果 $d_k=0:\lambda_k=q_{k-1}+\lambda_{k-1}$、$\eta_k=H(\lambda_k\|l_k\|\eta_{k-1}\|f_k)$、$\xi_k=\xi_{k-1}$ 和 $\zeta_k=\xi_{k-1}+q_k$。其中，$\eta_0=h(b_{i,j})$、$\lambda_1=1$、$\xi_0=0$ 和 $\zeta_0=0$。

那么在得到 $\{\lambda_\pi,\eta_\pi,\xi_\pi,\zeta_\pi\}$ 后，客户端将用 sig 验证 $R=\eta_\pi$，并且验证是否 $\xi_\pi=i-1$ 和 $\zeta_\pi=n-i$ 能够同时成立。如果三个值都通过了这个认证，那么 Ω_i (也是 $b_{i,j}$)和它的索引 i 的真实性可以被证实。

b）对于删除操作，客户需要验证 $h'\left(b_{i,j}\right)$。注意，$h'\left(b_{i,j}\right)$ 代表相同的块和副本，其 RST 的根被存储作为 Ω_i 中的第一个元组。客户端有足够的信息通过 $h\left(b_{i,j}\right)$、Ω_i 和 R 来验证 $h'\left(b_{i,j}\right)$。 对于插入操作，$h\left(b_{i,j}\right)$ 已经与 Ω_i 一起验证了，客户可以安全地计算新的 h_{i_new}，而不需要额外的验证。

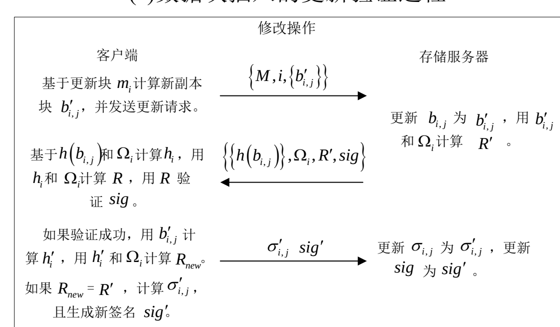
c）根据 RST 结构，客户端将根据 $h'\left(b_{i,j}\right)$ 计算 h_i'，然后根据 Ω_i 和 h_i' 计算 R_{new}，并将 R_{new} 与 R 进行比较。

以上验证成功后，客户端将更新块的数量 n，并计算同态标签 $\sigma_{i,j}'$ ($b_{i,j}'$ 的验证者)，存储在服务器上。

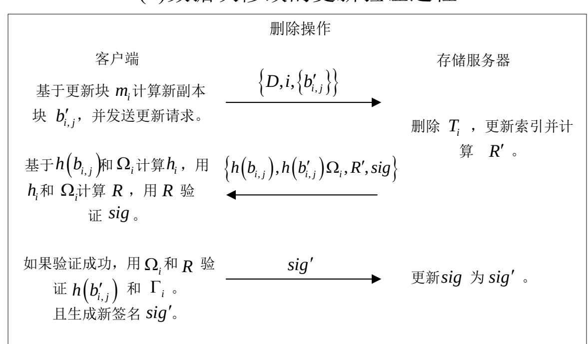
图 5 显示了数据更新验证的过程，其中分别包含了全块插入、删除和修改操作时的更新验证。

插入操作
客户端
存储服务器
基于更新块 m_i 计算新副本块 $b_{i,j}'$，并发送更新请求。
$\{I,i,\{b_{i,j}'\}\}$
用 $b_{i,j}'$ 创建一个新子树 T_i 更新叶节点的层级
基于 $h\left(b_{i,j}\right)$ 和 Ω_i 计算 h_i，用 h_i 和 Ω_i 计算 R，用 R 验证 sig。
$\{\{h\left(b_{i,j}\right)\},\Omega_i,sig\}$
如果验证成功，计算 $\sigma_{i,j}'$，且生成新签名 sig'。
$\sigma_{i,j}'$ sig'
更新 $\sigma_{i,j}$ 为 $\sigma_{i,j}'$，更新 sig 为 sig'。

(a)数据块插入的更新验证过程



(b)数据块修改的更新验证过程



(c)数据块删除的更新验证过程

图 5 数据块的更新和验证过程

3.4 抵御服务器攻击

本文验证方案中的标签数据都是通过密码哈希函数 H 来构建。为此，云存储中会存在以下类型的攻击：

a)重放攻击。当用户更新一个数据块后，会相应的计算一个标签 $H(i)$。如果存在一个恶意的存储服务器，其私自更改数据，并当审计者审计数据完整性时，将之前的标签 $H(i)$ 提交给审计者进行验证。

为此，本文在数据完整性验证之前，先通过 MHT 和每个副本的初始向量 $v_i=\{h(b_{i,1}),h(b_{i,2}),\cdots,h(b_{i,j})\}$，分别计算 $h\left(b_{i,j}\right)$ 和 $h'\left(b_{i,j}\right)$ 来计算 T_i 的新根 R_{new} 和旧根，然后计算标签 $\sigma_{i,j}'$ 来对标签进行了验证。

b)伪造攻击。在传统验证方案中，对于每个副本块 $b_{i,j}$，客户端将计算一个同态标签 $\sigma_{i,j}$。那么，此时恶意存储服务器能够伪造一个标签，例如 $\sigma_{i,j}=\left(H\left(b_{i,j}\right)\mu^{b_{i,j}}\right)^\alpha$ 和 $\sigma_{k,j}=\left(H\left(b_{k,j}\right)\mu^{b_{k,j}}\right)^\alpha$ 分别为两个同态标签。恶意服务器可以根据同态标签的运算规则，制造一个假标签 $\sigma'=\sigma_{i,j}\cdot\sigma_{k,j}=\left(H\left(b_{i,j}\right)\cdot H\left(b_{k,j}\right)\cdot\mu^{b_{i,j}+b_{k,j}}\right)^\alpha$。

同样，为了抵抗伪造攻击，本文对向量 $v_i=\{h(b_{i,1}),h(b_{i,2}),\cdots,h(b_{i,j})\}$ 进行了验证，结合索引信息就可以知道数据库的位置。为此，伪造签名不能通过审计者检查。

3.5 多副本公共审计的挑战和验证过程

在本文的自顶向下的分层设置中，验证者将需要 $H\left(b_{i,j}\right)$ 来验证审计方程，因为它不存储在多副本 MHT 中。这里本文讨论如何一次性对一组副本块进行验证。

a)产生挑战操作 $GenChallenge(Acc,pk,sig_{AUTH})$：第三方审计员 TPA 以给定准确度 Acc 生成挑战消息，并发送授权。挑战信息是 $\{sig_{AUTH},i,v_{i,j}\}_{i\in I}$，其中 sig_{AUTH} 用于授权，I 是为验证而选择的随机索引集合，$v_{i,j}$ 是用于 $b_{i,j}$ 的集成的随机数。

b)产生证据操作 $GenProof(pk,F,\Phi,chal)$：云服务器将首先验证 sig_{AUTH}。那么对于每个副本，它将计算 $\sigma_j=\prod_{i\in I}\sigma_{i,j}^{v_{i,j}}$ 和 $\mu_j=\sum_j v_{i,j}b_{i,j}$，并发送 $\left\{u_j,\sigma_j,\left\{H\left(b_{i,j}\right),h\left(b_{i,j}\right),\Omega_i\right\}_{i\in I},sig\right\}$ 到 TPA。

c)验证操作 $Verify(pk,P)$：由于验证者知道 RST 的结构，所以它将用 $\left\{h\left(b_{i,j}\right),\Omega_i\right\}$ 计算 R，并且对第 i 个选择的块验证 sig。此外，还需要通过验证是否 $h\left(b_{i,j}\right)=H\left(1\|l\left(b_{i,j}\right)\|H\left(b_{i,j}\right)\right)$ 成立来验证 $H\left(b_{i,j}\right)$ 的真实性，其中可以从 Ω_i 中第一个节点层级的 $l\left(h_i\right)$ 推断出

$l\left(b_{i,j}\right)$。例如，在图 3 中，如果设置复制品数量 $c=3$，那么 $l\left(h_1\right)=2$（h_1 是 Ω_2 中的第一个节点），$l\left(h_2\right)=2$，此时可以很容易地推导出 $l\left(b_{2,1}\right)=4$、$l\left(b_{2,2}\right)=4$ 和 $l\left(b_{2,3}\right)=3$。如果这个验证通过，TPA 将认为检索到的 $H\left(b_{i,j}\right)$ 是真实的，那么它可以通过验证以下等式来验证 c 个复制品：

$$e\left(\sigma_j,g\right)=e\left(\prod_i H\left(b_{i,j}\right)^{v_{i,j}}\mu^{\mu_j},v\right),\ j\in\left[1,c\right]。$$

如果这些等式成立，则验证将输出“接受”，否则输出“拒绝”。

3.6 安全性分析

本节对本文方法的安全性进行证明。本文方案的安全性主要是基于散列函数的抗碰撞性，双线性 Diffie-Hellman 问题的难度，以及所用签名方案的不可伪造性。下面对本文方案中数据更新认证的安全性假设进行证明。

假设：如果在服务器执行更新请求 $\left\{Type,i,\left\{b'_{i,j}\right\}\right\}$ 中，新数据内容或索引有任何错误，那么客户端验证将失败。

证明 服务器返回 RST 根 h_i 和它的 AAI Ω_i，并判断是否正确，否则 R 的验证将失败。

对于插入和修改操作，如果 $b'_{i,j}$ 被错误更新，那么由于散列函数 H 的抗碰撞性，那么 h'_i 和 R' 将会被错误地计算。根据 MHT 的性质，Ω_i 在整个更新期间内保持相同。由于客户有正确的 $b'_{i,j}$ 和 Ω_i，可以计算正确的 h'_i 和 R'。为此客户端的值 h'_i 和 R' 与反馈值不同，所以验证会失败。

对于删除操作，一旦此更新中有任何错误，返回的 $h\left(b'_{i,j}\right)$ 将不正确。由于 $h\left(b'_{i,j}\right)$ 包含在 $\Omega\left(b_{i,j}\right)$ 中，如果 $h\left(b'_{i,j}\right)$ 不正确，客户将能识别异常。

因此，通过本文的验证方案，客户端将能够检测到更新中由于意外或不诚实行为而导致的任何故障。

4 损坏定位与数据恢复

当发生数据损坏时，需要通过其他数据来进行数据恢复。在完整性审计中，如果审计结果为损坏，则需要对数据进行恢复。此时，用户向 CSP 发送定位请求信息，CSP 通过二分查找的方式来定位损坏数据块的位置。在找到损坏数据块的位置后，只要 j 个副本中有对应的数据块没有被损坏，则可以通过 Shamir 秘密共享机制来进行恢复。假设，数据块副本 $\tilde{F}_j$ 中存在数据损坏，数据恢复的流程如下描述：

a）用户向 CSP 发送数据恢复请求。

b）CCS 定位损坏数据，并通过完整性验证检测出对应的且完整的数据块 $b_{r_1,j},b_{r_2,j},\ldots,b_{r_k,j}$，并将其发送给用户。

c）用户根据 Shamir 秘密共享机制，在本地搜索出对应的 Shamir 参数 $x_{r_1},x_{r2},\ldots,x_{r_k}$。

d）用户对每个 Shamir 参数计算 Lagrange 插值多项式：$L_l(0)=\prod_{r_1\le l\le r_k,t\ne l}\frac{-x_t}{x_l-x_t}$，从而可以得到原始数据块

$$m_j=\sum_{l=r_1}^{r_k}L_l(0)b_{l,j}。$$

e）用户随机选择 $a_{k-1},\ldots,a_1\in Z_l$，并从 $\{x_1,x_2,\ldots,x_n\}$ 中选择 x_i，用来重新计算数据块 m_j 的第 i 个副本 $b_{i,j}=a_{k-1}x_i^{k-1}+\cdots+a_1x_i+a_{0j}$，$a_{0j}=m_j$。并将其发送给 CSP 替换掉遭受损坏的数据块。

5 实验及分析

5.1 实验环境

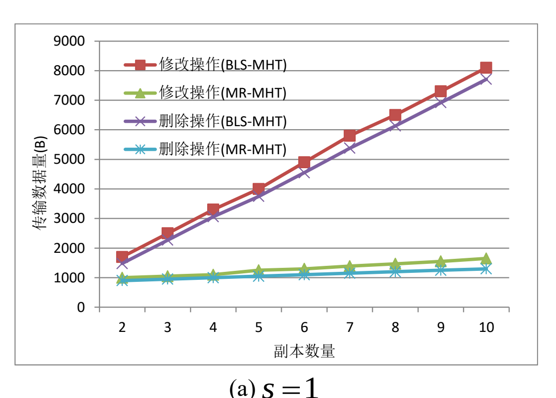
本文构建一个云计算环境用来进行实验。每个计算节点上安装了 Linux 操作系统。在虚拟化数据中心安装 Hadoop 以方便实现 MapReduce 编程模型和分布式文件系统[17]。此外，还安装了 OpenStack 开源云平台、负责全局管理、资源调度、任务分配以及与用户的交互。云平台共具有 36 个 CPU 核心，32 GB RAM 和 1 TB 存储的虚拟机。

5.2 性能评估

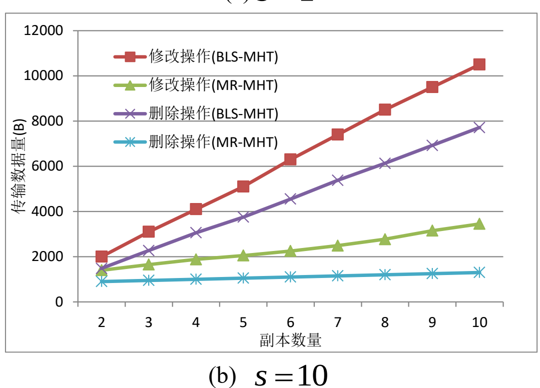
将本文提出的基于多副本 MHT 的数据验证方法(MR-MHT)与文献[7]提出的基于 BLS 签名和 MHT 的动态数据验证方案(BLS-MHT)，在更新验证过程的通信和存储成本方面进行比较。其中，BLS-MHT 是一种具有独立索引副本的动态公众审计的方案。使用 1GB 随机生成的数据集进行测试，副本为 $b_{i,j}=m_{i,j}+r_{i,j}$。BLS 参数选择为 80 位，即 G 的阶数长度为 160 位。另外，为了简化过程，设定每块的扇区数为 $s=1$ 和 $s=10$ 两种情况。

首先，本文测试了不同副本数量 c 和块大小 s 值下数据修改、插入和删除操作的数据传输开销，结果如图 6 所示。可以看出，BLS-MHT 的数据传输开销将随着数据副本数量的增加而迅速增加。对于块插入和修改，需要上传新的数据块。因此，总传输开销会随着的增加而增加。但 MR-MHT 的数据传输开销要显著小于 BLS-MHT。

在删除操作中，由于没有数据上传过程，所以在不同的 s 值下，两种方法的总传输开销保持不变。无论哪种情况，与 BLS-MHT 相比，MR-MHT 中更新验证的传输开销始终具有显著优势。



(a) $s=1$



(b) $s=10$

图 6 不同 s 时一个块验证更新的总通信

然后，对于动态完整性数据审计的存储开销和通信开销进行分析。虽然认证者总数保持不变，但本文中只有一个 MHT，而不是 BLS-MHT 中的 c 个 MHT。图 7 给出了 $s=10$ 时，不同副本数量下，在服务器端的额外存储开销，以支持公共审计和数据动态更新。可以看出，当存在多个副本时，MR-MHT 的额外存储成本会大大降低。

图 8 给出了 $s=10$ 时，多副本验证的通信开销。可以看到，MR-MHT 方案优于 BLS-MHT。另外，随着副本数量的增长，MR-MHT 中用于验证所有副本的通信开销与验证单个副本相当，而 BLS-MHT 的开销则成倍增长。例如，当 $c=5$ 时，使用 MR-MHT 验证所有 5 个副本的通信量比仅仅验证 1 个副本($c=1$)多 27%，而 BLS-MHT 却多了 390%。因此，MR-MHT 方案不仅对于动态数据的验证有效，而且在进行多次副本更新时也具有良好效果。

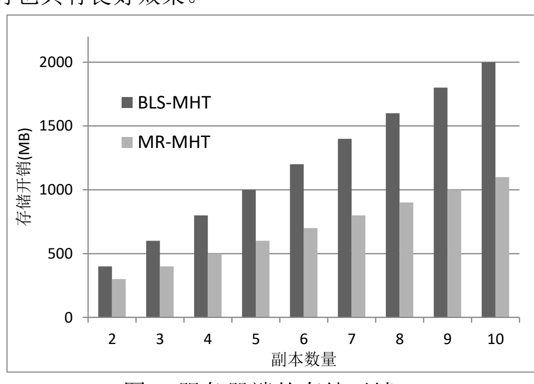


图 7 服务器端的存储开销。

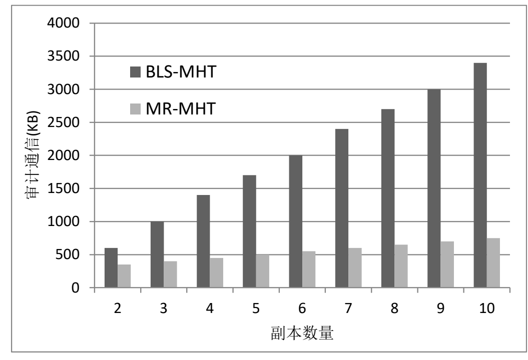


图 8 审计所有副本的通信开销。

从这些分析和实验结果可以看出，MR-MHT 方案在审计具有多个副本的云存储方面具有显著的优势。公开审计的执行不受数据内容的影响。因此，文件块大小，s 值和副本数量是影响整体性能的主要因素。

6 结束语

本文提出了一种基于多副本 MHT 的新型公共审计方案（MR-MHT）。引入了一种基于 MHT 的新型认证数据结构，树中节点的层级值以自顶向下的顺序生成，并将数据块的所有副本块组织成相同的副本子树。因此，该方案可以同时支持完全动态的数据更新、块索引验证和多个副本的高效验证。理论分析和实验结果表明，与现有的完整性验证方案相比，本文 MR-MHT 方案能够实现对具有多个副本的云数据集进行完整性验证，具有更少的通信和存储开销。

参考文献：

[1] 李谢华，刘婷，周茂仁. 云存储中基于多授权机构可撤销的 ABE 访问控制方法 [J]. 计算机应用研究, 2017, 34 (3): 897-902. (Li Xiehua, Liu Ting, Zhou Maoren. Multi-authority and revocable ABE scheme in cloud storage [J]. Application Research of Computers, 2017, 34 (3): 897-902.)

[2] 薛矛，薛巍，舒继武，等. 一种云存储环境下的安全存储系统 [J]. 计算机学报, 2015, 38 (5): 987-998. (Xue Mao, Xue Hao, Shu Jiwu, *et al.* A secure storage system over cloud storage environment [J]. Chinese Journal of Computers, 2015, 38 (5): 987-998.)

[3] 何凯，黄传河，王小毛，等. 云存储中数据完整性的聚合盲审计方法 [J]. 通信学报, 2015, 36 (10): 119-132. (He Kai, Huang Chuanhe, Wang Xiaomao, *et al.* Aggregated privacy-preserving auditing for cloud data integrity [J]. Journal on Communications, 2015, 36 (10): 119-132.)

[4] Dara U S, Chandra M S. Integrity verification in multiple cloud storage, using cooperative PDP method [J]. International Journal of Engineering Trends & Technology, 2013, 4 (9): 132-138.

[5] 李明富，陈立伟. 一种基于身份代理重加密的云数据共享方案 [J]. 湘潭大学自然科学学报, 2017, 39 (3): 75-79. (Li Mingfu, Chen Liwei. A secure cloud data sharing scheme from identity-based proxy re-encryption [J]. Natural Science Journal of Xiangtan University, 2017, 39 (3): 75-79.)

[6] Erway C, Küpçü A, Papamanthou C, *et al.* Dynamic provable data possession [J]. ACM Trans on Information & System Security, 2015, 17 (4): 15-22.

[7] Wang Qian, Wang Cong, Ren Kui, *et al.* Enabling public auditability and

data dynamics for storage security in cloud computing [J]. IEEE Trans on Parallel & Distributed Systems, 2011, 22 (5): 847-859.

[8] Liu Chang, Chen Jinjin, Yang Laurence T, *et al.* Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates [J]. IEEE Trans on Parallel & Distributed Systems, 2014, 25 (9): 2234-2244.

[9] 田晖，陈羽翔，黄永峰，等. 基于 Shamir 秘密共享的云端多副本审计 [J]. 华中科技大学学报：自然科学版, 2016, 44 (3): 77-82. (Tian Hui, Chen Yuxiang, Huang Yongfeng, *et al.* Multiple-replica auditing in clouds using Shamir secret sharing [J]. Journal of Huazhong University of Science and Technology: Nature Science Edition, 2016, 44 (3): 77-82.)

[10] 查雅行，罗守山，卞建超，等. 基于多分支认证树的多用户多副本数据持有性证明方案 [J]. 通信学报, 2015, 36 (11): 80-91. (Cha Yahang, Luo Shoushan, Bian Jianchao, *et al.* Multiuser and multiple-replica provable data possession scheme based on multi-branch authentication tree [J]. Journal on Communications, 2015, 36 (11): 80-91.)

[11] Curtmola R, Khan O, Burns R, *et al.* MR-PDP: multiple-replica provable data possession [C]// Proc of International Conference on Distributed Computing Systems. Piscataway: IEEE Press, 2008: 411-420.

[12] Kachkeev A, Esiner E, KüpçüA, *et al.* Energy efficiency in secure and dynamic cloud storage [C]// Proc of European Conference on Energy Efficiency in Large Scale Distributed Systems. Berlin: Springer, 2013: 125-130.

[13] Li Limin, Yang Yahui, Wu Zhonghai. FMR-PDP: flexible multiple-replica provable data possession in cloud storage [C]// Computers and Communications. : Piscataway, NJ: IEEE Press, 2017: 1115-1121.

[14] 沈丽敏，张福泰，孙银霞. 对一种无双线性配对的无证书签密方案的安全性分析 [J]. 密码学报, 2014, 1 (2): 146-154. (Shen Limin, Zhang Futai, Sun Yinxia. Security analysis of a certificateless signcryption scheme without bilinear pairing [J]. Journal of Cryptologic Reseatch, 2014, 1 (2): 146-154.)

[15] Kim S H, Kim Y, Kwon O, *et al.* Fully Batch processing enabled memory integrity verification algorithm based on Merkle tree [C]// Proc of International Workshop on Information Security Applications. Berlin: Springer, 2015: 386-398.

[16] 周彦伟，杨波，张文政. 不使用双线性映射的无证书签密方案的安全性分析及改进 [J]. 计算机学报, 2016, 39 (6): 1257-1266. (Zhou Yanwei, Yang Bo, Zhang Wenzheng. Security analysis and improvement of certificateless signcryption scheme without bilinear pairing [J]. Chinese Journal of Computers, 2016, 39 (6): 1257-1266.)

[17] 刘继华，强彦. 基于 Hadoop 分布式计算平台的磁流体动力学模型仿真研究 [J]. 计算机应用研究, 2017, 34 (5): 1353-1357. (Liu Jihua, Qiang Yan. Security analysis and simulation of magneto hydro dynamic model based on Hadoop distributed computing platform [J]. Application Research of Computers, 2017, 34 (5): 1353-1357.)